Proceedings of the 2005 IEEE
Engineering in Medicine and Biology 27th Annual Conference
Shanghai, China, September 1-4, 2005

Client Side Decompression Technique Provides Faster DNA Sequence Data Delivery

Fahim Sufi, Qiang Fang, Irena Cosic, Roy Ferguson,

School of Electrical and Computer Engineering,

RMIT University, City Campus, Melbourne, VIC 3001, Australia

Abstract— **DNA sequences are generally very long chains of sequentially linked nucleotides. There are four different nucleotides and combinations of these build the nucleotide information of sequence files contained in data sources. When a user searches for any sequence for an organism, a compressed sequence file can be sent from the data source to the user. The compressed file then can be decompressed at the client end resulting in reduced transmission time over the Internet. A compression algorithm that provides a moderately high compression rate with minimal decompression time is proposed in this paper. We also compare a number of different compression techniques for achieving efficient delivery methods from an intelligent genomic search agent over the Internet.**

I. Introduction

It is estimated that the number of available nucleotide bases nearly doubles every 14 months [1]. A typical single gene sequence comprises of many thousands of base pairs. For a whole genome, we need to consider millions and even billions of base pairs. Consequently, storage and communication of these ever-increasing DNA sequences is driving research into DNA sequence compression algorithms. In addition, compression techniques can be used to capture various properties and patterns of DNA sequences. Using compressed sequences pattern matching among DNA sequences can also become much faster [2]. It has been reported that some compression pattern matching algorithms [2] can run 10 times faster than the exact pattern matching algorithms, such as Agrep [3], on comparatively longer sequences.

Ref. [4, 5] describe several techniques for general purpose text compression. Lempel-Ziv based compression schemes [6, 7] are popular dictionary based algorithms.

Adjeroh et al. proposes the Burrows Wheeler Transform, suffix tree and suffix array based new offline dictionary compression scheme for DNA sequences [8]. Context tree weighting [9] is a data compression technique that employs context-based methods.

Generally, DNA sequences hold only four letters (A, T, G and C) and it is efficient to use only 2 bits per symbol [11], i.e., 00, 01, 10 and 11 is sufficient to represent the 4 letters. There are several DNA sequence compression algorithms that provide higher compression ratios resulting in less than 2 bits per symbol. This is possible because of the two main characteristic structures of sequences. One is reverse complement and the other is approximate repeat.

Regular compression programs like gzip fail to compress or compact DNA sequences because they in fact expand the file size by using more than 2 bits per symbol.

Biocompress-1 and Biocompress-2 [10] were the first compression schemes designed for DNA sequence compression based on an online dictionary dependent method. Unlike Biocompress-1, Biocompress-2 can also handle palindromes.

Both Biocompress-2 and GenCompress [11] are Lempel-Ziv based data compression algorithms [6] that search for both exact repeats and reverse complements. GenCompress also encodes approximate reverse complements using length, position and the error. The results from GenCompress can be used to form phylogenetic trees. Cfact [12] is another algorithm that is similar to Biocompress-2 except for its 2 pass behaviour. Only non-random repeat, which occurs due to some established biological phenomena [12, 13, 21], is utilized for Cfact compression. Biocompress-2 and GenCompress both use order-2 arithmetic but Cfact doesn't. There are other methods [14] combining Context Tree Weighting, which is an LZW-77 type algorithm and heuristics. Short repeats are encoded by context trees and larger repeats are encoded by LZ77. But these algorithms run considerably slower [15]. DNACompress [15] makes use of pattern hunter [16] and runs faster than other existing sequence compression tools.

In this paper, we use compression techniques for the novel purpose of sequence data delivery to the client. Existing DNA search engines do not utilise DNA sequence compression algorithms for client side decompression, i.e. where a compressed DNA sequence is decompressed at the client end for the benefit of faster transmission. Because most of the existing DNA sequence compression algorithms aim for higher compression ratios or pattern revealing, rather than client side decompression, their decompression times are longer than necessary. This makes these compression techniques unsuitable for the "on the fly" decompression. We use a compression technique designed for client side

0-7803-8740-6/05/$20.00 ©2005 IEEE.

decompression in order to achieve faster sequence data transmission to the client.

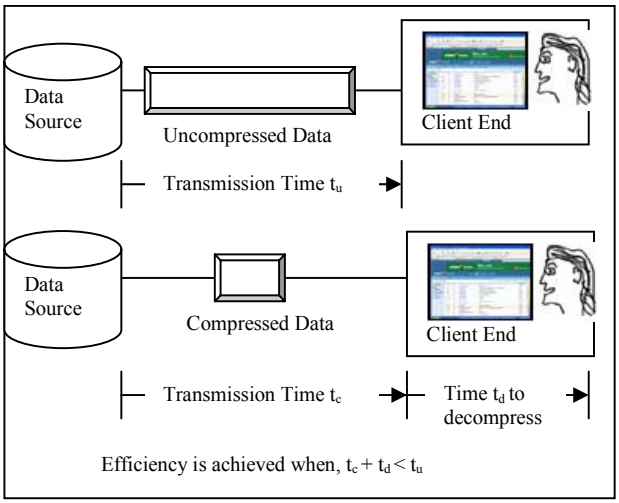

Fig. 1. Efficiency factor in client side decompression technique

If compressed sequence data is sent from the data source to be decompressed at the client end and the decompression time along with the compressed file transmission time is less than the transmission time for uncompressed data transfer from the source to the client, then efficiency is achieved. Fig. 1 illustrates the situation. Note that the sequence data should be kept pre-compressed within the data source.

A Sequence compression algorithm with reduced decompression time and moderately high compression rate is the preferred choice for efficient sequence data delivery with faster data transmission. As our target is to minimize decompression time, we use similar compression techniques to those used in [2], based on a "One Pass" approach, meaning, that the file is compressed or decompressed while reading it. Unlike "two pass" algorithms there is no need to re-read the input file. Our SeqDeliver compression technique is essentially a symbol substitution compression scheme that encodes the sequence by replacing four consecutive nucleotide sequences with ASCI characters.

SeqDeliver system encodes the sequence using the following steps:

1. Read the next four nucleotide codes from the DNA sequence and represent A, T, G and C by 0, 1, 2 and 3 respectively

2. Concatenate the resulting four digits of these base 4 (2 bit) numbers to give a single 8-bit number (one byte)

3. Write the byte obtained in step 2 to the compressed file.

4. Continue steps 1, 2 and 3 until the end of the DNA sequence is reached

As the highest four digit base 4 number (3333) is 255 in decimal, we can easily map the 4 digit sequence containing a combination of the letters A, T, G and C to a single byte. For an example, if we encounter a sequence "TTCC", when we represent the sequence by base 4 numbers it becomes 1133. In decimal this base 4 number is 147 and its corresponding ASCII code is ".

The decompression technique can be easily done in one pass by following steps:

1. We populate a 256 element array with all 4 character sequence combinations from AAAA to CCCC

2. We read a single byte from the compressed file,

3. We replace the single byte from the compressed file with the array element (Step 1) indexed by that byte (Step 2)

4. Continue steps 2 & 3 until the end of the compressed file is reached

LZW [6] based algorithms require a standard amount of time for finding repeats. Most of the existing sequence compression algorithms employ an LZW based algorithm for finding repeats and complemented palindromes. Thus, the minimal processing time required for an LZW based sequence compression algorithm is the time required for the LZW algorithm plus the remaining part of the algorithm without LZW. Chen et al. [15] showed that DNACompress is thought to be the fastest sequence compressor [15] by using PatternHunter [16] with some functionality of the LZW algorithm. We compare the decompression time of the original LZW based algorithm, DNACompress and our SeqDeliver technique to find the best solution for a client side decompression technique.

III. Result And Findings

Our technique, SeqDeliver, takes less time during decompression compared with many existing efficient algorithms at the cost of reduced compression ratio compared with existing sequence compression algorithms based on LZW. Due to the 4 to 1 mapping in the compression, the size of the compressed file will increase only linearly with the original file size at a constant compression ratio of 75%. We randomly selected some DNA sequences with varying length of nucleotides from NCBI's data source then compressed them with our technique, DNA Compress and the LZW based method respectively. We then decompressed them with the corresponding methods to record the decompression time taken by these methods for each of the sequences. Table I gives the detailed results from this comparison.

The results from Table I show SeqDeliver to be the best solution for client side decompression with the shortest and linearly increasing decompression time. However, SeqDeliver doesn't compress sequences as much as DNACompress for many of the cases in the compression ratio table of [15]. This is because SeqDeliver uses 2 bits to represent one nucleotide

In order to compare the overall performance, we conducted further studies involving sending actual sequence files of varying sizes (without compression) to measure the

calculated time (tu) needed for the transmission from the source to the destination. Then we compressed those files using both SeqDeliver and DNACompress, since DNACompress showed better compression ratio [15] in many cases over CTW+LZ and GenCompress. The total time t_t, defined as the sum of the compressed file transmission time (t_c) plus the client side decompression time (t_d), is measured by both these methods.

The results show that SeqDeliver is more efficient than DNACompress as it delivers decompressed sequence data much faster to the client. Table II shows the result of our comparisons for efficiency factors in the client side decompression technique. This result suggests that SeqDeliver is much more efficient than DNACompress as a tool to deliver sequences to the client. Further the results in Table II also show that the SeqDeliver based client side delivery method takes less than half of the time needed for delivery when employing no compression technique.

TABLE I
DECOMPRESSION TIME COMPARISON FOR DNA COMPRESS, LZW BASED SYSTEM AND SEQDELIVER

NCBI Accession ID	File Length	Decompression Time (Milliseconds)		
		DNA Compress	LZW Based System	SeqDeliver
AB099882	4242	320	69	20
AC006723	23764	521	339	89
AC006631	30767	1482	430	119
AC006131	36817	631	491	140
AC010642	45383	1001	631	179
AC004380	51065	411	710	190
AC006790	52139	1562	750	190
AC006733	58777	862	811	219
AC004131	77611	1912	1081	290
AC005212	79815	1812	1081	301
AC004763	87050	1262	1251	321
AC000002	93911	1772	1292	340
AC005333	94797	911	1291	359
AC000005	118159	3175	1612	439
AC000003	122228	2524	1723	451
AC005285	209071	1342	2943	770

NB: for compressed methods, the total time is defined as the sum of the transmission time plus the decompression time and for the uncompressed method, the total time is the transmission time only. All times are in millisecond. The transmission times are calculated by assuming a typical 50Kbps Internet connection and the comparison study was performed on a Pentium II 300 MHz computer with 256 Mb of RAM.

TABLE II
EFFICIENCY FACTORS FOR CLIENT SIDE DECOMPRESSION TECHNIQUE[a]

File Size (Bytes)	DNACompress (Milliseconds)	SeqDeliver (Milliseconds)	Uncompressed (Milliseconds)
5000	521.12	220	800
24000	1370.76	1049	3840
31000	2459.12	1359	4960
36000	1981.56	1580	5760
45000	2618.92	1979	7200
50000	2390.52	2190	8000
51000	3320.56	2230	8160
58000	2865.52	2539	9280
76000	4700.64	3330	12160
78000	4543.84	3421	12480
86000	4535.28	3761	13760
92000	5229.12	4020	14720
93000	4537.4	4079	14880
116000	7351.96	5079	18560
120000	6881.92	5251	19200
205000	9392.72	8970	32800
337000	92059.32	14732	53920
685000	68731.52	29983	109600

IV. DISCUSSION

SeqDeliver can compress sequence files to the baseline standard, which is 2 bits per nucleotide, with a fast decompression time (Table I). After compression with SeqDeliver, the compressed file becomes 25% of size of the original file, which requires less time for transmission from the source to the destination. When the compressed file reaches the client computer, SeqDeliver decompresses the file in one fifth of the time required for uncompressed file transmission (our experiment was conducted on sequences of size 5 kilobytes to 685 kilobytes). The ratio of decompression time to original transmission time of the uncompressed sequence file (t_d / t_u), reduces with increasing file size. This means our client side decompression technique with SeqDeliver is a better choice for larger sequence files. Our client side decompression technique can be implemented by a genome search agent and decompression time can be estimated by two empirical equations according to our experiments.

If, for a given computer with a Pentium II 300 MHz CPU and 256 Mb of RAM, a= 1E-13, b= 2E-07, c= 0.0087 and d= 91.276 then, for a given sequence length x, compression time y (in milliseconds) can be predicted by the equation:

$$y = ax3 + bx2 + cx + d \qquad (1)$$

The graph of compression time verus sequence file length is somewhat non-linear, because of the complexity that arises when the sequence length is not divisible by 4, which means that not all the bit pairs in the last byte of the compressed file may represent valid nucleotides. One solution to this problem

is to add an extra byte at the end of compressed file which is the count (1-4) of the number of valid nucleotides in the previous byte.

The decompression time increases linearly with file length. So where, for a given computer, m=0.0037 and c= 1.5084, for a given sequence length x, the decompression time y (in milliseconds) can be predicted by the equation:

$$y = mx + c \qquad (2)$$

In reality, sequence files maintained in a typical data sources [1] contain both nucleotide and non-nucleotide information. Till now our techniques compress only the nucleotide information. All our experiments for server side compression and client side decompression were done by compressing nucleotide information for different organisms [Table I, 15].

Our research on compressing the non-nucleotide portion of sequence files will result in a more usable and generalized compression technique for efficient sequence data delivery in future.

In summary, SeqDeliver combines moderate compression with reduced decompression time to achieve the best performance for client side sequence delivery compared with existing techniques. Its linearity in decompression time and close linearity in compression time make it an effective compression tool for commercial usage. Given, for a particular connection speed, the efficiency achieved using SeqDeliver, this compression technique is recommended for transmission of queried sequence files.

ACKNOWLEDGMENT

This research was supported and funded by school of electrical and computer engineering, RMIT University.

REFERENCES

[1] NCBI Database, Accessed December, 2004. World Wide Web URL: http://www.ncbi.nlm.nih.gov/Database/index.html

[2] Chen, L., Lu, S. and Ram J. 2004. “Compressed Pattern Matching in DNA Sequences”. Proceedings of the 2004 IEEE Computational Systems Bioinformatics Conference (CSB 2004)

[3] Wu, S. and Manber, U. 1992. “Agrep: a fast approximate pattern-matching tool” In Usenix Winter 1992 Technical Conference, pp. 153-162.

[4] Bell, T.C., Cleary, J.C., and Witten, I.H. 1990. Text Compression, Prentice Hall, Englewood Cliffs, NJ.

[5] Witten, I. H., Moffat, A. and Bell, T. C. 1999. Managing Gigabytes: Compressing and Indexing Documents and Images, Morgan Kaufman.

[6] Ziv, J. and Lempel, A. 1977. A universal algorithm for sequential data compression, IEEE Trans. Inform. Theory. 23(3):337 – 343.

[7] Lempel, A. and Ziv, J. 1978. Compression of individual sequences via variable-rate coding, IEEE Trans. Inform. Theory, IT-24:530-536.

[8] Adjeroh, D., Zhang, Y., Mukherjee, A., Powell, M. and Bell T. 2002. "DNA Sequence Compression using the Burrows-Wheeler Transform”. Proceedings of the IEEE Computer Society Bioinformatics Conference (CSB’02)

[9] Willems, F. M. J., Shtarkov, Y. M. and Tjalkens, T. J. 1995. The context-tree weighting method: basic properties. IEEE Trans. Inform. Theory, IT-41, 653-664.

[10] Grumbach, S. and Tahi, F. 1994. A New Challenge for Compression Algorithms: Genetic Sequences. Information Processing & Management. 30:875–886.

[11] Chen, X., Kwong, S. and Li, M. December 1999. A Compression Algorithm for DNA sequences and Its Applications in Genome Comparison. In Proc. of the 10th Workshop on Genome Informatics (GIW’99). Pages 52–61.

[12] Rivals, E., Delahaye, J. P., Dauchet, M. and Delgrange, O. 1996. “A Guaranteed Compression Scheme for Repetitive {DNA} Sequences”. In the proceedings of Data Compression Conference.

[13] Rivals, E. and Dauchet, M. 1997. "Fast Siscerning Repeats in {DNA} Sequences with a Compression Algorithm." Proceedings Genome Informatics

[14] Matsumoto, T., Sadakane, K. and Imai, H. 2000. Biological sequence compression algorithms, Genome Informatics Workshop, Universal Academy Press, pp. 43–52.

[15] Chen, X., Li, M., Ma, B. and J. Tromp. 2002. "DNACompress: fast and effective DNA sequence compression". Bioinformatics. 18: 1696-1698.

[16] Ma, B., Tromp, J. and Li, M. 2002. "PatternHunter -- faster and more sensitive homology search". Bioinformatics. 18: 440-445.

[17] Allison, L., Edgoose, T., Dix, T. I. 28 June - 1 July 1998. Compression of Strings with Approximate Repeats. Intelligent Systems in Mol. Biol. ISMB98, Montreal, pp8–16.

[18] Sato, H., Yoshioka, T., Konagaya, A. and Toyoda, T. 2001. “DNA Data Compression in the Post Genome Era”. Proceedings Genome Informatics 2001

[19] Powell, D. R., Dowe, D. L., Allison, L. and Dix, T. I. 1998. "Discovering Simple {DNA} Sequences by Compression". Proceedings of Pacific Symposium on Biocomputing '98"

[20] Tabus, I., Korodi, G. and Rissanen, J. 2003. “DNA Sequence Compression Using the Normalized Maximum Likelihood Model for Discrete Regression.” Data Compression Conference DCC'2003, Snowbird, Utah, March 24-27, 2003.

[21] Rivals, E., Delgrange, O., Delahaye, J. P., Dauchet, M., Delorme, M. O., Henaut, A. and Ollivier, E. 1997. "Detection of Significant Patterns by Compression Algorithms: the Case of Approximate Tandem Repeats in DNA Sequences". Computer Applications in the Biosciences